PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-171667
(43)Date of publication of application : 26.06.1998

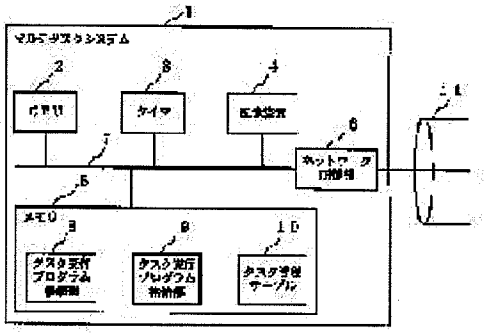
(51)Int.Cl. G06F 9/46

(21)Application number : 08-333667
(22)Date of filing : 13.12.1996

(71)Applicant : CHOKOSOKU NETWORK COMPUTER GIJUTSU KENKYUSHO:KK
(72)Inventor : ABE MUTSUMI

(54) TASK MANAGEMENT METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To eliminate the execution of tasks in a range exceeding the system processing capability by deciding whether the delay can be assured for a requested task to be reserved and a reserved task and then accepting the reservation of the requested task if its delay is assured.
SOLUTION: A memory 5 includes a task reception program storage part 8, a task execution program storage part 9 and a task management table 10 which stores the task information to manage the tasks which require the real- time properties. Then it is decided whether the delay can be assured for a requested task to be reserved and a reserved task, and the reservation is accepted for the requested task if its delay can be assured. At the same time, the tasks set in an executable state where their reservation are accepted and the events occurred are executed in response to their priority. Then other tasks are executed within a cycle of a relevant task despite the presence of an event and a high priority level.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（A）　(11)特許出願公開番号

特開平10－171667

(43)公開日　平成10年(1998)６月26日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｇ０６Ｆ　9/46	３４０	Ｇ０６Ｆ　9/46	３４０Ｂ ３４０Ｅ

審査請求　有　　請求項の数３　ＯＬ　（全 10 頁）

(21)出願番号　特願平8－333667

(22)出願日　平成８年(1996)12月13日

特許法第30条第１項適用申請有り　1996年10月25日　社団法人電子情報通信学会発行の「電子情報通信学会技術研究報告　信学技報Ｖｏｌ．96　Ｎｏ．327」に発表

(71)出願人　394025577
株式会社超高速ネットワーク・コンピュータ技術研究所
東京都港区虎ノ門五丁目２番６号

(72)発明者　阿部　睦
東京都港区虎ノ門５丁目２番６号　株式会社超高速ネットワーク・コンピュータ技術研究所内

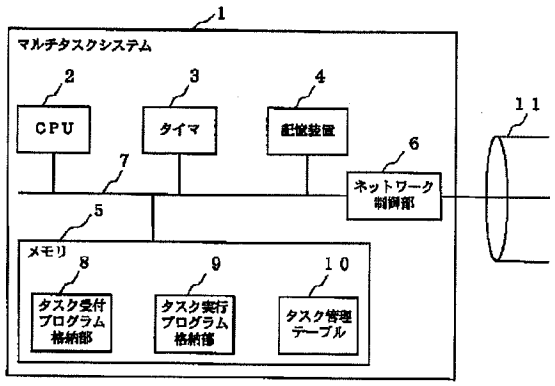
(74)代理人　弁理士　山川　政樹

(54)【発明の名称】　タスク管理方法

(57)【要約】

【課題】　タスクの遅延保証と効率的なタスクの実行を実現する。

【解決手段】　遅延保証が必要な各タスクについて、周期、要求処理時間及び許容遅延時間からなるタスク情報を設定し、要求処理時間と許容遅延時間の和が小さい順にタスクの優先度を高く設定する。タスクの予約が要求されたときは、このタスクと予約済みのタスクのタスク情報に基づき、要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断し、遅延保証が可能な場合にタスクの予約を受理する。予約が受け付けられイベントが発行された実行可能状態のタスクは、優先度に応じて実行され、またイベントが存在して優先度が高くても、そのタスクの周期内であれば、他のタスクが実行される。



1

【特許請求の範囲】

【請求項１】　複数のタスクを実行するマルチタスクシステムにおいて前記タスクを管理するタスク管理方法であって、

遅延保証が必要な各タスクについて、実行の周期、実行に必要とされる要求処理時間及び許容される遅延時間からなるタスク情報を設定し、

前記要求処理時間と許容遅延時間との和が小さい順にタスクの優先度を高く設定し、

遅延保証が必要なタスクの予約が要求されたときは、このタスクと予約済みのタスクのタスク情報に基づき、要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断して、遅延保証が可能な場合にタスクの予約を受理し、

予約済みの各タスクについては、そのタスクに対するイベントが発行され、前回の実行から周期以上の時間が経過し、且つそのタスクより優先度の高いタスクが実行されていないときに、該当タスクを実行することを特徴とするタスク管理方法。

【請求項２】　請求項１記載のタスク管理方法において、

周期に対する要求処理時間の割合を要求されたタスクと予約済みのタスクについて求めて、これらの加算値を求めると共に、優先度の一番低いタスクの要求処理時間と許容遅延時間の和に対する他のタスクの最大実行回数分の要求処理時間の和を求めて、これを優先度の一番低いタスクの許容遅延時間から減算して減算値を求め、

前記加算値が１以上で、かつ減算値が０以上であれば、要求されたタスクと受付済みのタスクの遅延保証が可能と判断して、要求されたタスクの予約を受理することを特徴とするタスク管理方法。

【請求項３】　請求項１記載のタスク管理方法において、

遅延保証を必要としないタスクについて、所望の周期を設定すると共に、この周期内で割り当てたい時間を要求処理時間、周期と要求処理時間の差を許容遅延時間として設定し、

遅延保証を必要とするタスクを受け付ける前に、前記遅延保証を必要としないタスクを予約しておくことを特徴とするタスク管理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、マルチタスクシステムに係り、特にプロセッサ資源の管理と遅延保証機能を有するタスク管理方法に関するものである。

【０００２】

【従来の技術】コンピュータ間通信において、動画や音声といったリアルタイム情報の通信を行いたいという要求は近年、一段と増している。動画や音声を扱うリアルタイム通信では、通信の品質が問題となるため、サービス品質の保証が必要となる。このようなサービス品質の保証を実現する手段として、ＡＴＭ（Asynchronous Transfer Mode）等の帯域設定の可能なネットワークや、ＸＴＰ（Xpress Transport Protocol）等のサービス品質制御機能を持つプロトコルが提案されている。

2

【０００３】しかし、ネットワークで帯域設定を行ったり、プロトコルに品質制御のための機能を設けても、端末側でそれに対応できていなければアプリケーション全体としての品質は保証されない。よって、サービス品質の保証はネットワークのみならず端末をも含めたシステム全体で保証する必要がある。ところで、リアルタイム通信端末では、例えば相手側の端末から動画や音声といった情報を受信するタスク、受信した情報を表示するタスク、相手側の端末に対して情報を送信するタスク等の複数のタスクを実行しなければならない。したがって、この端末は、リアルタイム処理を必要とする複数のタスクを実行するマルチタスクシステムとなり、このようなマルチタスクシステムにおいて、サービス品質の保証が必要となる。

【０００４】そこで、このようなマルチタスクシステムにおいて、サービス品質を保証する技術として、（１）ＯＳ（Operating System）のタスク優先度設定機能を用いて、動画や音声といった情報を扱うリアルタイム性を要求されるタスクの優先度を他のタスクより高く設定して、遅延要求の厳しいタスクの処理を優先させるという手法が一般的に使用されている。また、サービス品質を保証する他の技術として、（２）実行終了目標時刻が設定されたタスクについて、その実行終了目標時刻が近づくほど優先度を上げて優先的に処理することにより、タスク処理を実行終了目標時刻までに完了させるという手法が提案されている（特開昭６２－２８４４３７号公報）。また、他の技術として、（３）映像データや音声データ等といった周期的に発生するデータを処理するタスクに対して、周期的に実行割当を行うことにより対応するという手法が提案されている（河内谷他、「連続メディアのＱＯＳ制御のためのＯＳサポート」、情報処理学会コンピュータシステムシンポジウム、１９９４）。

【０００５】

【発明が解決しようとする課題】しかし、上記従来技術（１）、（２）では、優先度の高いタスクから実行するため、優先度の低いタスクの遅延要求（タスクが要求する許容可能な遅延時間）を満たせなくなってしまうという問題点があった。また、従来技術（１）、（２）、（３）では、タスクが起動されたとしても処理すべきデータが存在しなければ、プロセッサがデータ待ち等のウエイト状態に置かれるため、他のタスクが実行可能であってもプロセッサが割り当てられないことになり、処理効率が低下してしまうという問題点があった。さらに、従来技術（１）、（２）、（３）では、タスクの受付の制限がないので、遅延要求を維持することのできるタス

ク量を超えた（つまり、処理能力を超えた）タスクをシステムが実行しようとしてしまい、いずれか又はすべてのタスクの遅延要求を満たせなくなってしまうという問題点があった。

【０００６】本発明は、上記課題を解決するためになされたもので、タスクの遅延保証と効率的なタスクの実行を実現することができるタスク管理方法を提供することを目的とする。

【０００７】

【課題を解決するための手段】本発明のタスク管理方法は、請求項１に記載のように、遅延保証が必要な各タスクについて、実行の周期、実行に必要とされる要求処理時間及び許容される遅延時間からなるタスク情報を設定し、要求処理時間と許容遅延時間との和が小さい順にタスクの優先度を高く設定し、遅延保証が必要なタスクの予約が要求されたときは、このタスクと予約済みのタスクのタスク情報に基づき、要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断して、遅延保証が可能な場合にタスクの予約を受理し、予約済みの各タスクについては、そのタスクに対するイベントが発行され、前回の実行から周期以上の時間が経過し、且つそのタスクより優先度の高いタスクが実行されていないときに、該当タスクを実行するようにしたものである。このように、遅延保証が必要なタスクの予約が要求されたときは、このタスクと予約済みのタスクのタスク情報に基づき、要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断して、遅延保証が可能な場合にタスクの予約を受理する。そして、予約済みの各タスクの実行は、実行したいタスクに対してイベントを発行することで行なわれる。イベントの発行はそのタスクの実行を要求したアプリケーションや他のタスクから発行することが可能である。イベントが発行されないタスクは待ち状態となり、イベントが発行されている他のタスクが実行される。タスクの実行基準は、優先度を基準にして優先度の高いタスクから実行される。ただし、イベントが発行されていても、前回のタスクの実行開始からの時間がそのタスクの周期より短い場合は、そのタスクは待たされて、他のタスクが実行される。他のタスクの実行中であっても、それより高い優先度を持つタスクに対してイベントが発行され、且つ前回の実行からの時間がそのタスクの周期を超えている場合は、高い優先度を持つタスクを実行する。

【０００８】また、請求項２に記載のように、周期に対する要求処理時間の割合を要求されたタスクと予約済みのタスクについて求めて、これらの加算値を求めると共に、優先度の一番低いタスクの要求処理時間と許容遅延時間の和に対する他のタスクの最大実行回数分の要求処理時間の和を求めて、これを優先度の一番低いタスクの許容遅延時間から減算して減算値を求め、加算値が１以上で、かつ減算値が０以上であれば、要求されたタスクと受付済みのタスクの遅延保証が可能と判断して、要求されたタスクの予約を受理するようにしたものである。このように上記加算値と減算値を求めることにより、予約が要求されたタスクと受付済みのタスクの遅延保証が可能かどうかを容易に判断することができる。

【０００９】また、請求項３に記載のように、遅延保証を必要としないタスクについて、所望の周期を設定すると共に、この周期内で割り当てたい時間を要求処理時間、周期と要求処理時間の差を許容遅延時間として設定し、遅延保証を必要とするタスクを受け付ける前に、遅延保証を必要としないタスクを予約するようにしたものである。このように遅延保証を必要としないタスクについて、周期、要求処理時間及び許容遅延時間を設定することにより、以後の遅延保証が必要なタスクの受付は、遅延保証を必要としないタスクの割当分を差し引いた状態で行なわれる。

【００１０】

【発明の実施の形態】

実施の形態の１．次に、本発明について図面を参照して説明する。図１は本発明の第１の実施の形態となるタスク管理方法を用いるマルチタスクシステムのブロック図である。マルチタスクシステム１は、ＣＰＵ２、タイマ３、例えばハードディスク装置等の記憶装置４、メモリ５、ネットワーク１１とのインタフェースとなるインタフェース制御部６を備えている。そして、ＣＰＵ２、タイマ３、記憶装置４、メモリ５、ネットワーク制御部６は内部バス７で接続されている。

【００１１】タイマ３は、設定された一定時間までの計時を繰り返すものであり、設定時間ごとにＣＰＵ２に割り込みをかけることにより、設定時間の経過をＣＰＵ２に通知することが可能である。記憶装置４にはタスクプログラム等が記録されており、必要に応じてメモリ５に読み出される。

【００１２】また、本発明のタスク管理方法を実現するにはＣＰＵ制御が必要であり、タスク管理の機能をＯＳ（Operating System）に組み込む必要がある。そこで、メモリ５上には、後述するタスク受付処理をＣＰＵ２に実行させるためのタスク受付プログラムが格納されるタスク受付プログラム格納部８と、同様にタスク実行処理をＣＰＵ２に実行させるためのタスク実行プログラムが格納されるタスク実行プログラム格納部９が存在する。そして、これらプログラムを格納部８、９に格納することにより、本発明のタスク管理方法の機能がＯＳに組み込まれる。

【００１３】さらに、メモリ５上には、リアルタイム性が要求されるタスク、すなわち遅延時間を小さくする必要のあるタスク（以下、このようなタスクを遅延保証が必要なタスクと呼ぶ）を管理するために、これら各タスクの情報を格納しておくためのタスク管理テーブル１０が存在する。図２はタスク管理テーブル１０の１例を示

す図である。タスク管理テーブル１０は、タスク情報管理テーブル２１とタスク実行管理テーブル２２とから構成される。

【００１４】タスク情報管理テーブル２１は、後述するタスク受付処理で受け付けられた遅延保証が必要なタスクに関する情報を管理するテーブルである。このタスク情報管理テーブル２１に格納される情報としては、タスク識別子格納部２１ａ、タスクポインタ格納部２１ｂ、周期格納部２１ｃ、要求処理時間格納部２１ｄ、許容遅延時間格納部２１ｅにそれぞれ格納されるタスク識別子ＩＤ、タスクポインタＰ、周期Ｓ、要求処理時間Ｔ、許容遅延時間Ｈがあり、これらの情報がタスクごとに格納される。なお、図２では、横１列が１つのタスクに関する情報を示している。

【００１５】タスク識別子ＩＤは、各タスクを識別するためのものであり、タスクポインタＰは、記憶装置４あるいはメモリ５上に存在する各タスクのプログラムの位置を指し示すものである。周期Ｓは、各タスクにおける実行開始時点から次の実行開始時点までの時間である。よって、あるタスクが実行されると、設定された周期Ｓの間、このタスクが新たに実行されることはない。

【００１６】要求処理時間Ｔは、各タスクの実行に要する処理時間である。なお、この要求処理時間Ｔは、予め設定される値なので、実際のタスクは要求処理時間Ｔより前に終了することもあり得る。

【００１７】許容遅延時間Ｈは、周期Ｓで示される時間内でタスクが実行待ち状態でいられる時間である。あるタスクに対してイベントが発行された状態で前回の実行開始時点からの経過時間が周期Ｓを超えたり、タスク処理が中断したりすると、このタスクに遅延が生じる。このような遅延の発生は、許容遅延時間Ｈを消費することを意味する。したがって、この許容遅延時間Ｈが全て消費される前にタスク処理が終了しなければならない。そして、周期Ｓ、要求処理時間Ｔ、許容遅延時間Ｈは、Ｓ≧Ｔ＋Ｈの関係にある。

【００１８】次に、タスク実行管理テーブル２２は、受け付けられたタスク、すなわちタスク情報管理テーブル２１に登録されたタスクの実行時に使用される情報を管理するテーブルである。このタスク実行管理テーブル２２に格納される情報としては、タスク識別子格納部２２ａ、実行識別フラグ格納部２２ｂ、イベントカウンタ格納部２２ｃ、周期カウンタ格納部２２ｄにそれぞれ格納されるタスク識別子ＩＤ、実行識別フラグＦ、イベントカウンタＥＣ、周期カウンタＳＣがあり、これらの情報がタスクごとに格納される。

【００１９】実行識別フラグＦは、実行中または実行を中断しているタスクを識別するためのものである。この実行識別フラグＦは、各タスクが実行中または実行中断状態ならばセットされ（例えば、「１」となる）、タスクの実行が終了した場合はリセットされる（例えば、「０」となる）。イベントカウンタＥＣは、各タスクに対して発行されたイベント数を示すものである。このイベントカウンタＥＣは、タスクに対してイベントが発行されると１加算され、対応するタスクが実行されると１減算される。

【００２０】周期カウンタＳＣは、各タスクの実行開始時点からの経過時間を示すものである。この周期カウンタＳＣは、タスクの実行開始でリセットされ、その後はタイマ３の値に対応した値をとる。

【００２１】次に、遅延保証が必要なタスクの予約について説明する。あるタスクの実行に際しては、このタスクの実行が別のタスクの許容遅延時間の保証を乱してはならないという制約がある。また、前述のように、タスク処理は、許容遅延時間が全て消費される前に終了しなければならない。逆に言うと、許容遅延時間内に終了できるのであれば、タスク処理は中断しても構わない。そして、あるタスクが実行されると、設定された周期の間、このタスクが新たに実行されることはない。

【００２２】これらのことから、以下の制約が導きだされる。

（１）優先度は、要求処理時間と許容遅延時間との和によって決まり、この和が大きいタスクは処理の優先度が下がる。

（２）優先度の高いタスクの要求処理時間が優先度の低いタスクの許容遅延時間内に収まらなければならない。

この制約を満たせば、各タスクの許容遅延時間を保証しつつタスク処理の実行が可能となる。したがって、このような制約を満たすかどうかで、新たなタスク要求を受理すべきかどうかを判断することができる。

【００２３】図３はこのタスク受付処理を説明するためのフローチャート図である。例えばアプリケーションからのシステムコールにより、タスクの予約が要求されると、ＣＰＵ２は、タスク受付プログラム格納部８に格納されたプログラムに従って以下のような処理を実行する。

【００２４】最初に、ＣＰＵ２は、周期に対する要求処理時間の割合を既に受け付けられている各タスクについて求めると共に、同様の割合を受付要求が発生したタスクについて求め、これらを加算して加算値Ａｄｄを求める。

【００２５】例えば、周期がＳ１、要求処理時間がＴ１、許容遅延時間がＨ１に設定された図４（ａ）に示すタスクＡ１、周期がＳ２、要求処理時間がＴ２、許容遅延時間がＨ２に設定された図４（ｂ）に示すタスクＡ２、周期がＳ３、要求処理時間がＴ３、許容遅延時間がＨ３に設定された図４（ｃ）に示すタスクＡ３のうち、タスクＡ１、タスクＡ２が既に受け付けられたタスクとしてタスク管理テーブル１０に登録されているとすると、ＣＰＵ２は、タスク情報管理テーブル２１に登録された情報から、タスクＡ１の周期Ｓ１に対する要求処理

時間Ｔ１の割合Ｔ１／Ｓ１を求め、同様にタスクＡ２の周期Ｓ２に対する要求処理時間Ｔ２の割合Ｔ２／Ｓ２を求める。

【００２６】また、アプリケーションから予約が要求さ＊れたタスクＡ３の周期Ｓ３に対する要求処理時間Ｔ３の割合Ｔ３／Ｓ３を求める。これにより、加算値Ａｄｄは次式となる。

$$Add=(T1/S1)+(T2/S2)+(T3/S3) \quad \cdots(1)$$

【００２７】続いて、ＣＰＵ２は、優先度の一番低いタスクの要求処理時間＋許容遅延時間に対する他のタスクの最大実行回数分の要求処理時間を各タスクについて求めた後に、これらの和を求め、算出した和を優先度の一番低いタスクの許容遅延時間から減算して減算値Ｓｕｂを求める。優先度は、要求処理時間Ｔと許容遅延時間Ｈとの和によって決まり、この和が小さいものほど優先度が高いタスクとなる。これにより、図４では、タスクＡ１が優先度の一番低いタスクとなる。

【００２８】優先度の一番低いタスクＡ１の要求処理時間Ｔ１＋許容遅延時間Ｈ１中に他のタスクを最大何回実※※行できるかは、そのタスクの周期によって決まる。例えば、タスクＡ２の場合、この最大実行回数は、（Ｔ１＋Ｈ１）／Ｓ２となる。よって、優先度の一番低いタスクの要求処理時間＋許容遅延時間に対する他のタスクの最大実行回数分の要求処理時間とは、この最大実行回数に各タスクの要求処理時間を掛けたものとなり、タスクＡ２の場合は、Ｔ２×（Ｔ１＋Ｈ１）／Ｓ２となり、タスクＡ３の場合は、Ｔ３×（Ｔ１＋Ｈ１）／Ｓ３となる。

【００２９】そして、これらの和を優先度の一番低いタスクＡ１の許容遅延時間Ｈ１から減算するので、減算値Ｓｕｂは次式となる。

$$Sub=H1-[\{T2\times(T1+H1)/S2\}+\{T3\times(T1+H1)/S3\}] \quad \cdots(2)$$

【００３０】こうして、ステップ１０１におけるタスク時間計算が終了する。なお、本実施の形態では、３つのタスクで計算したが、４つ以上のタスクでも同様に計算すればよい。例えば、タスクＡ１、Ａ２、Ａ３・・・Ａｎ-1 （周期Ｓｎ-1 、要求処理時間Ｔｎ-1 、許容遅延時間★Ｈｎ-1 ）が予約済みのタスクで、タスクＡｎ （周期Ｓｎ 、要求処理時間Ｔｎ 、許容遅延時間Ｈｎ ）の予約要求が発生したとすると、求める加算値Ａｄｄは次式となる。

$$Add=(T1/S1)+(T2/S2)\cdots+(Tn-1/Sn-1)+(Tn/Sn) \quad \cdots(3)$$

【００３１】また、タスクＡ１の優先度が一番低いとす☆ ☆ると、求める減算値Ｓｕｂは次式となる。

$$Sub=H1-[\{T2\times(T1+H1)/S2\}+\{T3\times(T1+H1)/S3\}\cdots+\{Tn-1\times(T1+H1)/Sn-1\}+\{Tn\times(T1+H1)/Sn\}] \quad \cdots(4)$$

【００３２】次に、ＣＰＵ２は、これらの算出した値から受付要求が発生したタスクが受付可能かどうかを判定する（ステップ１０２）。ステップ１０１で算出した加算値Ａｄｄが割合基準値（＝１）を超えているか（Ａｄｄ＞１）、又は減算値Ｓｕｂが負の値であれば（Ｓｕｂ＜０）、要求されたタスクと受付済みのタスクの遅延保証が不可能と判断して受付不許可とし、受付できなかった旨をアプリケーションに返答して（ステップ１０３）、受付処理を終了する（ステップ１０６）。

【００３３】このようにＯＳ側から予約の不受理が通知された場合、アプリケーション側では、品質を調整して再度予約するか予約を断念するかの判断を行う。また、加算値Ａｄｄが割合基準値を超えておらず（Ａｄｄ≦１）、かつ減算値Ｓｕｂが０以上であれば（Ｓｕｂ≧０）、要求されたタスクと受付済みのタスクの遅延保証が可能と判断して受付許可とし、受付要求されたタスクＡ３の情報をメモリ５のタスク管理テーブル１０に書き込む（ステップ１０４）。続いて、受付できた旨をアプリケーションに返答して（ステップ１０５）、受付処理を終了する（ステップ１０６）。

【００３４】なお、受付要求が発生したタスクＡ３の周期Ｓ３、要求処理時間Ｔ３、許容遅延時間Ｈ３、タスクポインタＰ３は、このタスク予約を要求したアプリケーションに設定されており、上記システムコールによってＯＳに渡され、受付が許可されると、タスク識別子ＩＤ３が付与されてタスク情報管理テーブル２１に格納される。

【００３５】次に、タスクの実行管理処理について説明する。図５、図６はこのタスク実行管理処理を説明するためのフローチャート図である。遅延保証が必要なタスクが実行可能となる条件としては、そのタスクに対してイベントが発行されており、かつ前回のタスクの実行からそのタスクの周期以上の時間が経過していることが条件なので、タスクの実行管理はイベントが発行された時点とタスクの時間処理の時点で行えばよいことになる。

【００３６】まず、タスクに対してイベントが発行されたときのタスクの実行管理を図５を参照して説明する。イベントとは、マルチタスクシステム１内での動画データの入出力やネットワーク１１からの動画パケットの受信あるいはネットワーク１１への動画パケットの送信等の要求である。

【００３７】このようなイベントが発生すると、ＣＰＵ

２は、タスク実行管理テーブル２２に格納された、このイベントに対応するタスクのイベントカウンタＥＣを１加算する（ステップ２０１）。次いで、ＣＰＵ２は、タスク実行管理テーブル２２に登録されている各タスクの情報により、新規に動作可能なタスクがあるかどうかを判定する（ステップ２０２）。

【００３８】動作可能なタスクとは、イベントが発行されている即ちイベントカウンタＥＣの値が１以上で、かつ前回の実行開始時点からそのタスクの周期が経過している則ち周期カウンタＳＣの値がそのタスクの周期Ｓ以上の値となっているタスクである。ステップ２０２において、動作可能なタスクがない場合はイベント処理を終了する（ステップ２０３）。

【００３９】また、動作可能なタスクが存在する場合、ＣＰＵ２は、タスク実行管理テーブル２２に登録されている各タスクの実行識別フラグＦにより、動作中のタスクがあるかどうかを判定する（ステップ２０４）。動作中のタスクがない場合は、ステップ２０２で見つけた動作可能なタスクを実行すると同時に、タスク実行管理テーブル２２に登録された、このタスクの周期カウンタＳＣをリセットする（ステップ２０５）。

【００４０】また、ステップ２０４において動作中のタスクが存在する場合は、この動作中のタスクとステップ２０２で見つけた動作可能なタスクの優先度を比較する（ステップ２０６）。動作中のタスクの方が優先度が高い場合は、イベント処理を終了し（ステップ２０７）、動作可能なタスクの方が優先度が高い場合は、動作中のタスク処理を中断して、動作可能なタスクを実行すると同時に、タスク実行管理テーブル２２に登録された、このタスクの周期カウンタＳＣをリセットする（ステップ２０５）。

【００４１】なお、実行が中断されたタスクは、後述する時間処理において、このタスクより優先度の高いタスクが存在しなければ、処理が再開される。次に、タスクの時間処理が発生したときのタスクの実行管理を図６を参照して説明する。この時間処理の実行タイミングの１例としては、タスクの周期Ｓと比べて十分に短い１ミリ秒や百マイクロ秒といった一定値が設定されたタイマ３により、一定時間ごとに発生するＣＰＵ２への割り込み時点がある。

【００４２】タスクの時間処理が発生すると、ＣＰＵ２は、タスク実行管理テーブル２２に登録されている各タスクの周期カウンタＳＣの値をタイマ３の値に基づいて再設定する（ステップ３０１）。この再設定は、周期カウンタＳＣの現在の値にタイマ３の値を加算することで行われる。

【００４３】続いて、ＣＰＵ２は、動作中のタスクが存在するかどうかを判定する（ステップ３０２）。動作中のタスクが存在する場合は、このタスクの要求処理時間が終了しているかどうかを判定する（ステップ３０３）。そして、動作中のタスクの要求処理時間が終了していない、即ちテーブル２２に格納された、このタスクの周期カウンタＳＣの値がテーブル２１に格納された要求処理時間Ｔ以上でないならば、時間処理を終了する（ステップ３０４）。

【００４４】また、動作中のタスクの要求処理時間が終了している、即ち周期カウンタＳＣの値が要求処理時間Ｔ以上ならば、タスクの実行が終了したものとして、タスク実行管理テーブル２２に格納された、このタスクのイベントカウンタＥＣを１減ずる（ステップ３０５）。

【００４５】また、ステップ３０２において動作中のタスクが存在しない場合あるいはステップ３０５の処理が終了した場合は、動作可能なタスクが存在するかどうかを判定する（ステップ３０６）。

【００４６】動作可能なタスクとは、前述のようにイベントカウンタＥＣの値が１以上で、周期カウンタＳＣの値がそのタスクの周期以上となっているタスクである。このようなタスクが複数ある場合は、それらの中で優先度が最も高いタスクを実行すべきタスクとして、これを実行すると同時に、タスク実行管理テーブル２２に登録された、このタスクの周期カウンタＳＣをリセットする（ステップ３０７）。そして、ステップ３０６において動作可能なタスクが存在しない場合は、時間処理を終了する（ステップ３０８）。

【００４７】図７は以上のような実行管理処理の１例を示す図である。ここでは、周期がＳ４、要求処理時間がＴ４、許容遅延時間がＨ４に設定された図７（ａ）に示すタスクＡ４、周期がＳ５、要求処理時間がＴ５、許容遅延時間がＨ５に設定された図７（ｂ）に示すタスクＡ５のうち、タスクＡ４の優先度が高い。

【００４８】このようなタスクＡ４、Ａ５が受け付けられ、各タスクに対してイベントが発行されると、上記イベント処理により、優先度の高いタスクＡ４がまず実行され（図７（ｃ））、続いてタスクＡ４の要求処理時間Ｔ４が経過した時間ｔ１に達すると、上記時間処理により、タスクＡ５が実行される（図７（ｄ））。

【００４９】このタスクＡ５の実行中にも時間処理が繰り返されているので、時間ｔ２にてタスクＡ４の周期カウンタが周期Ｓ４以上になると、タスクＡ５の処理が中断されて優先度の高いタスクＡ４が実行される。そして、タスクＡ５の処理は、タスクＡ４の要求処理時間Ｔ４が経過した時間ｔ３から再開される。

【００５０】以上のように、本発明では、予約が要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断して、遅延保証が可能な場合にタスクの予約を受理するので、システムの処理能力を超えてタスクを実行することがなくなる。

【００５１】そして、予約が受け付けられイベントが発行された実行可能状態のタスクを優先度に応じて実行し、かつイベントが存在して優先度が高くても、そのタ

スクの周期内であれば、他のタスクを実行するので、各タスクを適切に実行することができる。また、イベントが発行されていないタスクを実行することがないので、優先度の高いタスクにおけるデータ待ちをなくすことができ、プロセッサ資源を無駄に消費することがなくなり、システムの有効利用が可能となる。

【００５２】なお、本実施の形態では、タスクの時間処理の実行タイミングとして、一定値が設定されたタイマ３によるＣＰＵ２への割り込み時点を例に挙げているが、図５のステップ２０５及び図６のステップ３０７においてタスクを実行させるときに、このタスクの要求処理時間の残り時間をタイマ３に設定することにより（新たに実行するタスクであれば、要求処理時間をそのまま設定し、中断状態にあったタスクであれば、要求処理時間から中断した時間を引いた時間を設定する）、タイマ３からＣＰＵ２に割り込みがかかった時点と、図５のステップ２０１の処理が終了した時点とを時間処理の実行タイミングとしてもよい。

【００５３】この実行タイミングによれば、イベント発行時と各タスクの動作停止時にのみ時間処理が行われるので、これより短い一定時間ごとに時間処理を実行する上記の例に比べて、時間処理によるオーバーヘッドを軽減することができる。

【００５４】実施の形態の２．実施の形態の１では、遅延保証が必要なタスクだけがタスク管理テーブル１０に登録されるようになっており、ＣＰＵ２は、タスク管理テーブル１０に登録されたタスクを優先的に処理するため、遅延保証を必要としないタスクが存在しても、このタスクが処理されないことがあり得る。そこで、このような点に対応するためには、実施の形態の１とは異なる処理が必要となる。

【００５５】図８は本発明の他の実施の形態となるタスク管理方法を示すフローチャート図である。本実施の形態においても、マルチタスクシステムの構成、タスク受付処理、タスク実行管理処理は実施の形態の１と同様であり、本実施の形態の処理は、遅延保証が必要なタスクの予約の前に実施される。

【００５６】まず、記憶装置４から遅延保証を必要としないタスクに割り当てたいタスクの割合を読み出す（ステップ４０１）。この割合は、元となる時間とその時間内で割り当てたい時間とからなる。そして、元となる時間を遅延保証を必要としないタスクの周期とし、割り当てたい時間を要求処理時間として、周期と要求処理時間の差を許容遅延時間とする。

【００５７】続いて、遅延保証が必要なタスクと同様に受付要求を発して、これらの情報をタスク管理テーブル１０へ登録させる（ステップ４０２）。このように遅延保証を必要とするタスクを受け付ける前に、遅延保証を必要としないタスクを擬似的にタスク管理テーブル１０に登録すると、以後の遅延保証が必要なタスクの受付は、遅延保証を必要としないタスクの割当分を差し引いた状態で行なわれるので、遅延保証を必要としないタスクにも一定の割合でタスクが実行されることを保証することができる。

【００５８】なお、ステップ４０２でタスク管理テーブル１０に登録されるタスクは、遅延保証を必要としない実際のタスクではなく、テーブル１０に登録するための疑似的なタスクなので、実行後直ちに終了するタスクでよい。つまり、遅延保証を必要としない実際のタスクは、タスク管理テーブル１０に基づく管理外で実行される。ただし、イベントが発行され、前回の実行から周期以上の時間が経過し、かつ疑似タスクより優先度の高いタスクが存在すれば、遅延保証を必要としないタスクの処理は中断されて優先度の高いタスクが実行される。

【００５９】

【発明の効果】本発明によれば、請求項１に記載のように、予約が要求されたタスクと予約済みのタスクの遅延保証が可能かどうかを判断して、遅延保証が可能な場合にタスクの予約を受理するので、システムの処理能力を超えてタスクを実行することがなくなる。そして、予約が受け付けられイベントが発行された実行可能状態のタスクを優先度に応じて実行し、かつイベントが存在して優先度が高くても、そのタスクの周期内であれば、他のタスクを実行するので、各タスクを適切に実行することができる。また、イベントが発行されていないタスクを実行することがないので、優先度の高いタスクにおけるデータ待ちをなくすことができ、プロセッサ資源を無駄に消費することがなくなる。その結果、全てのタスクの遅延要求を満たすことができ、かつシステムの利用効率を向上させることができる。

【００６０】また、請求項２に記載のように、加算値と減算値を求めて、これらに基づいて判断することにより、予約が要求されたタスクと受付済みのタスクの遅延保証が可能かどうかを容易に判断することができる。

【００６１】また、請求項３に記載のように、遅延保証を必要としないタスクについて、周期、要求処理時間及び許容遅延時間を設定することにより、以後の遅延保証が必要なタスクの受付は遅延保証を必要としないタスクの割当分を差し引いた状態で行なわれるので、遅延保証を必要とするタスクだけが実行されることを防ぎ、遅延保証を必要としないタスクの実行も保証することができる。

【図面の簡単な説明】

【図１】　本発明の第１の実施の形態となるタスク管理方法を用いるマルチタスクシステムのブロック図である。

【図２】　タスク管理テーブルの１例を示す図である。

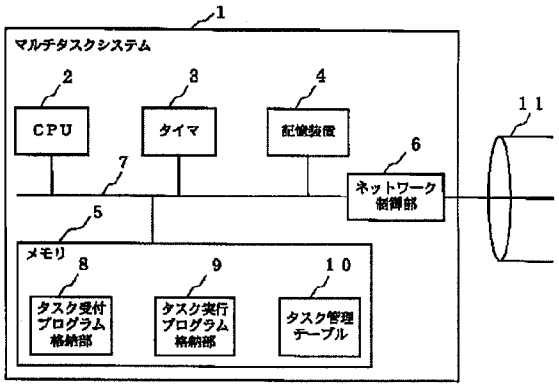
【図３】　タスク受付処理を説明するためのフローチャート図である。

【図４】　タスク受付処理を説明するための図である。

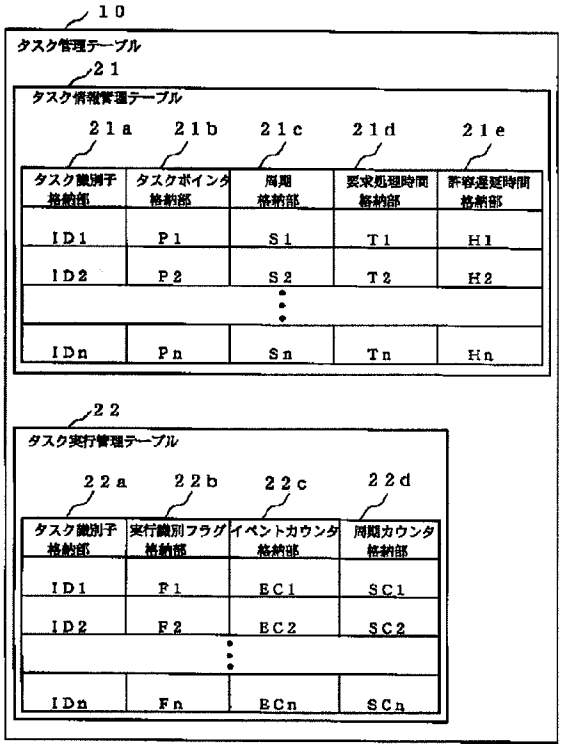
【図５】　タスク実行管理処理を説明するためのフローチャート図である。
【図６】　タスク実行管理処理を説明するためのフローチャート図である。
【図７】　タスク実行管理処理の１例を示す図である。
【図８】　本発明の他の実施の形態となるタスク管理方法を示すフローチャート図である。　＊

＊【符号の説明】
１…マルチタスクシステム、２…ＣＰＵ、３…タイマ、４…記憶装置、５…メモリ、６…インタフェース制御部、７…内部バス、１０…タスク管理テーブル、１１…ネットワーク、２１…タスク情報管理テーブル、２２…タスク実行管理テーブル。

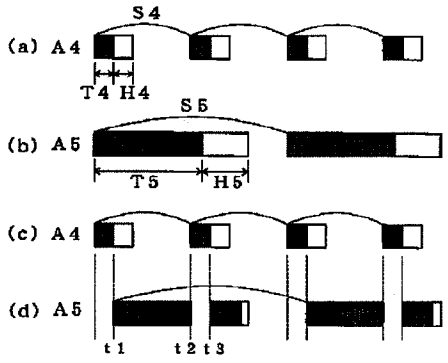
【図１】



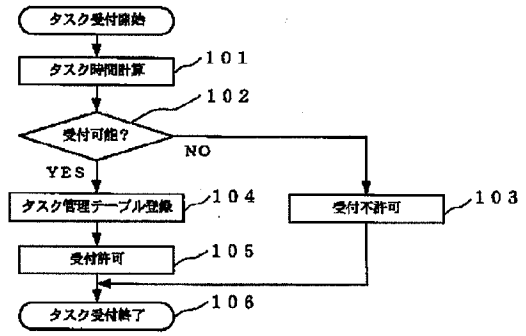
【図２】



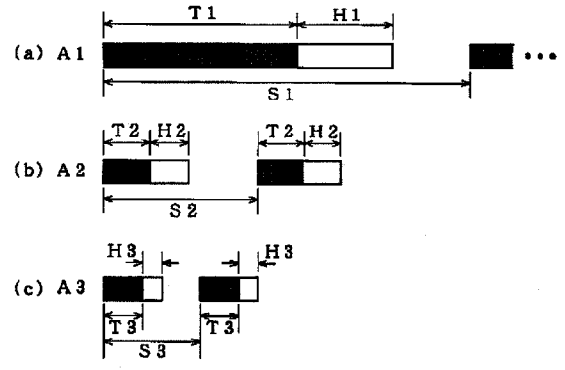
【図７】



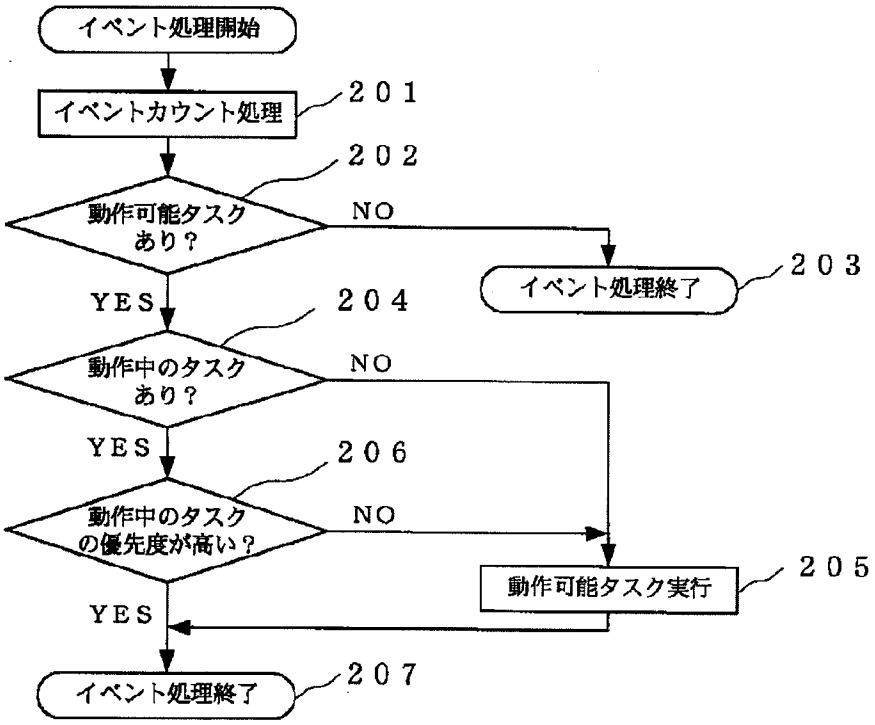
【図３】



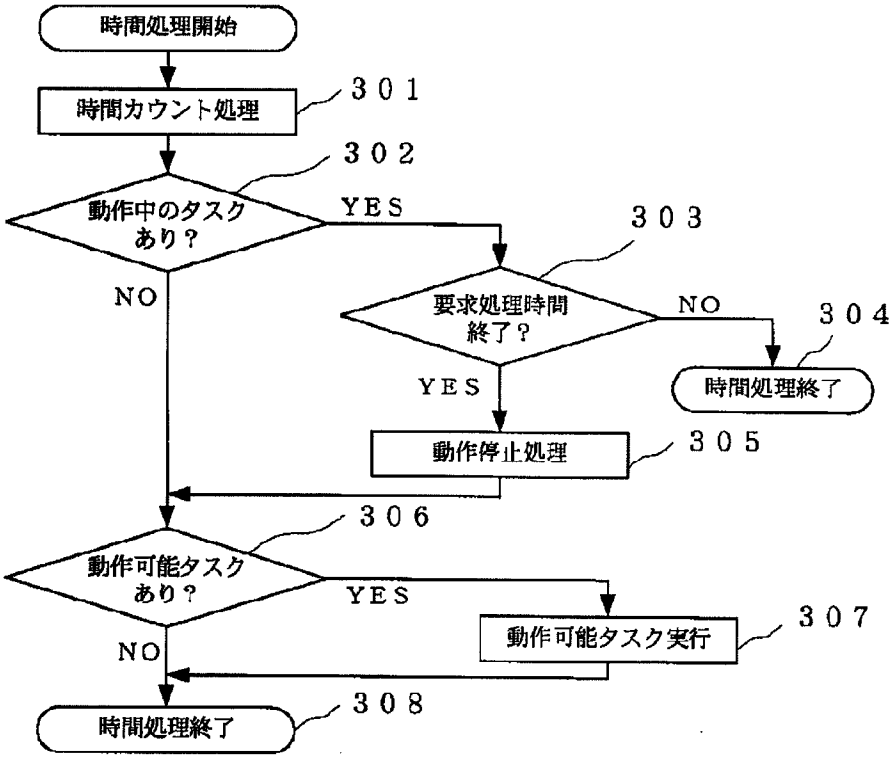
【図４】



【図5】


イベント処理開始
イベントカウント処理
201
202
動作可能タスク
あり？
NO
イベント処理終了
203
YES
204
動作中のタスク
あり？
NO
YES
206
動作中のタスク
の優先度が高い？
NO
動作可能タスク実行
205
YES
イベント処理終了
207

【図6】


時間処理開始
時間カウント処理
301
302
動作中のタスク
あり？
YES
303
NO
要求処理時間
終了？
NO
304
YES
時間処理終了
動作停止処理
305
306
動作可能タスク
あり？
YES
動作可能タスク実行
307
NO
時間処理終了
308

【図８】